

Mayet-Godowski Hilbert Lattice Equations

Norman D. Megill*

Boston Information Group, 19 Locke Ln., Lexington, MA 02420, USA

Mladen Pavičić†

Physics Chair, Faculty of Civil Engineering, University of Zagreb, Croatia

Several new results in the field of Hilbert lattice equations based on states defined on the lattice as well as novel techniques used to arrive at these results are presented. An open problem of Mayet concerning Hilbert lattice equations based on Hilbert-space-valued states is answered.

PACS numbers: 02.10.-v, 03.65.Fd, 03.67.Lx

Keywords: Hilbert space, Hilbert lattice, strong state, quantum logic, quantum computation, Godowski equations, Mayet-Godowski equations, orthomodular lattice

I. INTRODUCTION

When we deal with quantum systems and its theoretical Hilbert space, we know that there is a Hilbert lattice that is isomorphic to the set of subspaces of any infinite-dimensional Hilbert space and that we can establish a correspondence between elements of the lattice and solutions of a Schrödinger equation that corresponds to such a Hilbert space. Therefore we might attempt to arrive at an algebra which would enable us to introduce quantum problems in a would-be quantum computer in the same way in which we can introduce Boolean algebra problem into a classical computer. The gain is exponential—*any* quantum problem could be solved in a polynomial time.

But there is an essential problem here. Any Hilbert lattice is a structure based on first-order predicate calculus, and we simply cannot have a constructive procedure to introduce statements like *there is* or *for all* into a quantum computer. What we might do instead is to find classes of polynomial lattice equations that can serve in place of quantified statements. How far we have advanced down this road we recently reviewed in Refs. [1] and [2], and in this paper we consider recent results we obtained for particular classes of such equations—Mayet-Godowski ones.

In 1985, René Mayet [3] described a new equational variety of lattices, which he called OM_S^* , that included all Hilbert lattices and were included in a related variety of equations found by Radosław Godowski [4] in 1981. However, it was not known whether the new variety was smaller than Godowski's (i.e., whether its equations were independent from Godowski's).

Recently, the authors showed [2] that Mayet's variety is indeed strictly included in Godowski's. In order to achieve this result, several new algorithms had to be developed to find counterexamples efficiently, to generate new equations in the family that were violated by the counterexamples, and to prove that the new equations were independent from every equation in the infinite family found by Godowski. This paper describes these algorithms, which were incorporated into the computer programs that found this result.

In the last section, we show the solution to an open problem posed by Mayet [5], related to another new family of equations that he found, based on strong sets of Hilbert-space-valued states.

II. DEFINITIONS FOR LATTICE STRUCTURES

We briefly recall the definitions we will need. For further information, see Refs. [2, 6, 7, 8].

Definition II.1 [9] *A lattice is an algebra $L = \langle \mathcal{L}_O, \cap, \cup \rangle$ such that the following conditions are satisfied for any $a, b, c \in \mathcal{L}_O$:*

$$\begin{array}{ll} a \cup b = b \cup a & a \cap b = b \cap a \\ (a \cup b) \cup c = a \cup (b \cup c) & (a \cap b) \cap c = a \cap (b \cap c) \\ a \cap (a \cup b) = a & a \cup (a \cap b) = a \end{array}$$

*Electronic address: nm@alum.mit.edu; URL: <http://www.metamath.org>

†Electronic address: pavicic@grad.hr; URL: <http://m3k.grad.hr/pavicic>

Theorem II.2 [9] *Binary relation \leq defined on L as*

$$a \leq b \stackrel{\text{def}}{\iff} a = a \cap b \quad \text{or as} \quad a \leq b \stackrel{\text{def}}{\iff} b = a \cup b \quad (1)$$

is a partial ordering

Definition II.3 [10] *An ortholattice (OL) is an algebra $\langle \mathcal{L}_O, ', \cap, \cup, 0, 1 \rangle$ such that $\langle \mathcal{L}_O, \cap, \cup \rangle$ is a lattice with unary operation $'$ called orthocomplementation which satisfies the following conditions for $a, b \in \mathcal{L}_O$ (a' is called the orthocomplement of a):*

$$a \cup a' = 1, \quad a \cap a' = 0 \quad (2)$$

$$a \leq b \Rightarrow b' \leq a' \quad (3)$$

$$a'' = a \quad (4)$$

Definition II.4 [11, 12] *An orthomodular lattice (OML) is an OL in which the following condition holds:*

$$a \leftrightarrow b = 1 \iff a = b \quad (5)$$

where $a \leftrightarrow b = 1 \stackrel{\text{def}}{\iff} a \rightarrow b = 1 \ \& \ b \rightarrow a = 1$, where $a \rightarrow b \stackrel{\text{def}}{=} a' \cup (a \cap b)$

Definition II.5 [13] *We say that a and b commute in OML, and write aCb , when either of the following equivalent equations hold:*

$$a = ((a \cap b) \cup (a \cap b')) \quad (6)$$

$$a \cap (a' \cup b) \leq b \quad (7)$$

Definition II.6 [22] *An orthomodular lattice which satisfies the following conditions is a Hilbert lattice, \mathcal{HL} .*

1. Completeness: *The meet and join of any subset of an \mathcal{HL} exist.*
2. Atomicity: *Every non-zero element in an \mathcal{HL} is greater than or equal to an atom. (An atom a is a non-zero lattice element with $0 < b \leq a$ only if $b = a$.)*
3. Superposition principle: *(The atom c is a superposition of the atoms a and b if $c \neq a$, $c \neq b$, and $c \leq a \cup b$.)*
 - (a) *Given two different atoms a and b , there is at least one other atom c , $c \neq a$ and $c \neq b$, that is a superposition of a and b .*
 - (b) *If the atom c is a superposition of distinct atoms a and b , then atom a is a superposition of atoms b and c .*
4. Minimal length: *The lattice contains at least three elements a, b, c satisfying: $0 < a < b < c < 1$.*

Definition II.7 *A state (also called probability measures or simply probabilities [14, 15, 16, 17]) on a lattice \mathcal{L} is a function $m : \mathcal{L} \rightarrow [0, 1]$ such that $m(1) = 1$ and $a \perp b \Rightarrow m(a \cup b) = m(a) + m(b)$, where $a \perp b$ means $a \leq b'$.*

Lemma II.8 *The following properties hold for any state m :*

$$m(a) + m(a') = 1 \quad (8)$$

$$a \leq b \Rightarrow m(a) \leq m(b) \quad (9)$$

$$0 \leq m(a) \leq 1 \quad (10)$$

$$m(a_1) = \dots = m(a_n) = 1 \iff m(a_1) + \dots + m(a_n) = n \quad (11)$$

$$m(a_1 \cap \dots \cap a_n) = 1 \Rightarrow m(a_1) = \dots = m(a_n) = 1 \quad (12)$$

Definition II.9 *A nonempty set S of states on \mathcal{L} is called a strong set of states if*

$$(\forall a, b \in L)(\exists m \in S)((m(a) = 1 \Rightarrow m(b) = 1) \Rightarrow a \leq b). \quad (13)$$

Theorem II.10 [7] *Every Hilbert lattice admits a strong set of states.*

III. DEFINITIONS OF EQUATIONAL FAMILIES RELATED TO STATES

First we will define the family of equations found by Godowski, introducing a special notation for them. These equations hold in any lattice admitting a strong set of states and thus, in particular, any Hilbert lattice. [7]

Definition III.1 *Let us call the following expression the Godowski identity:*

$$a_1 \stackrel{\gamma}{\equiv} a_n \stackrel{\text{def}}{=} (a_1 \rightarrow a_2) \cap (a_2 \rightarrow a_3) \cap \cdots \cap (a_{n-1} \rightarrow a_n) \cap (a_n \rightarrow a_1), \quad n = 3, 4, \dots \quad (14)$$

Theorem III.2 *Godowski's equations [4]*

$$a_1 \stackrel{\gamma}{\equiv} a_3 = a_3 \stackrel{\gamma}{\equiv} a_1 \quad (15)$$

$$a_1 \stackrel{\gamma}{\equiv} a_4 = a_4 \stackrel{\gamma}{\equiv} a_1 \quad (16)$$

$$a_1 \stackrel{\gamma}{\equiv} a_5 = a_5 \stackrel{\gamma}{\equiv} a_1 \quad (17)$$

...

hold in all ortholattices, OL's, with strong sets of states. An OL to which these equations are added is a variety smaller than OML.

We shall call these equations n -Go (3-Go, 4-Go, etc.). We also denote by n GO (3GO, 4GO, etc.) the OL variety determined by n -Go and call it the n GO law.

Next, we define a generalization of this family, first described by Mayet. [3] These equations also hold in all lattices admitting a strong set of states, and in particular in all HLs.

Definition III.3 *A Mayet-Godowski equation (MGE) is an equality with $n \geq 2$ conjuncts on each side:*

$$t_1 \cap \cdots \cap t_n = u_1 \cap \cdots \cap u_n \quad (18)$$

where each conjunct t_i (or u_i) is a term consisting of either a variable or a disjunction of two or more distinct variables:

$$t_i = a_{i,1} \cup \cdots \cup a_{i,p_i} \quad \text{i.e. } p_i \text{ disjuncts} \quad (19)$$

$$u_i = b_{i,1} \cup \cdots \cup b_{i,q_i} \quad \text{i.e. } q_i \text{ disjuncts} \quad (20)$$

and where the following conditions are imposed on the set of variables in the equation:

1. All variables in a given term t_i or u_i are mutually orthogonal.
2. Each variable occurs the same number of times on each side of the equality.

We will call a lattice in which all MGEs hold an MGO; i.e., MGO is the class (equational variety) of all lattices in which all MGEs hold.

The following three theorems about MGEs and MGOs are proved in Ref. [2].

Theorem III.4 *A Mayet-Godowski equation holds in any ortholattice \mathcal{L} admitting a strong set of states and thus, in particular, in any Hilbert lattice.*

Theorem III.5 *The family of all Mayet-Godowski equations includes, in particular, the Godowski equations [Eqs. (15), (16), ...]; in other words, the class MGO is included in n GO for all n .*

Theorem III.6 *The class MGO is properly included in all n GOs, i.e., not all MGE equations can be deduced from the equations n -Go.*

Definition III.7 *A condensed state equation is an abbreviated version of an MGE constructed as follows: all (orthogonality) hypotheses are discarded, all meet symbols, \cap , are changed to $+$, and all join symbols, \cup , are changed to juxtaposition.*

For example, the 3-Go equation can be expressed as: [2]

$$\begin{aligned} a \perp d \perp b \perp e \perp c \perp f \perp a &\Rightarrow \\ (a \cup d) \cap (b \cup e) \cap (c \cup f) &= (d \cup b) \cap (e \cup c) \cap (f \cup a), \end{aligned} \quad (21)$$

which, in turn, can be expressed by the condensed state equation

$$ad + be + cf = db + ec + fa. \quad (22)$$

The one-to-one correspondence between these two representations of an MGE should be obvious.

IV. CHECKING n -GO EQUATIONS ON FINITE LATTICES

For the general-purpose checking of whether an equation holds in a finite lattice, the authors have primarily used a specialized program, `latticeg.c`, that is specialized to check an equation provided by the user against a list of Greechie diagrams (OMLs) provided by the user. This program has been described in Ref. [18]. While it has proved essential to our work, a drawback is that the run time increases quickly with the number of variables in and size of the input equation, making it impractical for huge equations.

But there is another limitation in principle, not just in practice, for the use of the `latticeg.c` program. In our work with MGEs, we are particularly interested in those lattices having no strong set of states but on which all of the successively stronger n -Gos pass, for all n less than infinity. This would prove that any MGE failing in that lattice is independent from all n -Gos and thus represents a new result. The `latticeg.c` program can, of course, check only a finite number of such equations, and when n becomes large the program is too slow to be practical. And in any case, it cannot provide a proof, but only evidence, that a particular lattice does not violate n -Go for any n .

Both of these limitations are overcome by a remarkable algorithm based on dynamic programming, that was suggested by Brendan McKay. This algorithm was incorporated into a program, `latticego.c`, that is run against a set of lattices. No equation is given to the program; instead, the program tells the user the first n for which n -Go fails or whether it passes for all n . The program runs very quickly, depending only on the size of the input lattice, with a run time proportional to the fourth power of the lattice size, rather than increasing exponentially with n as with the `latticeg.c` that checks against arbitrary equations.

To illustrate the algorithm, we will consider the specific case of 7-Go. From this example, the algorithm for the general case of n -Go will be apparent. We consider 7-Go written in the following equivalent form: [7]

$$(a_1 \rightarrow a_2) \cap (a_2 \rightarrow a_3) \cap (a_3 \rightarrow a_4) \cap (a_4 \rightarrow a_5) \cap (a_5 \rightarrow a_6) \cap (a_6 \rightarrow a_7) \cap (a_7 \rightarrow a_1) \leq a_1 \rightarrow a_7 \quad (23)$$

We define intermediate “operations” E_1, \dots, E_6 along with a predicate which provides the the answer:

$$\begin{aligned} E_1(a_1, a_2) &= a_1 \rightarrow a_2 \\ E_2(a_1, a_2, a_3) &= E_1(a_1, a_2) \cap (a_2 \rightarrow a_3) \\ E_3(a_1, a_2, a_3, a_4) &= E_2(a_1, a_2, a_3) \cap (a_3 \rightarrow a_4) \\ E_4(a_1, a_2, a_3, a_4, a_5) &= E_3(a_1, a_2, a_3, a_4) \cap (a_4 \rightarrow a_5) \\ E_5(a_1, a_2, a_3, a_4, a_5, a_6) &= E_4(a_1, a_2, a_3, a_4, a_5) \cap (a_5 \rightarrow a_6) \\ E_6(a_1, a_2, a_3, a_4, a_5, a_6, a_7) &= E_5(a_1, a_2, a_3, a_4, a_5, a_6) \cap (a_6 \rightarrow a_7) \\ \text{answer}(a_1, a_7) &= (E_6(a_1, a_2, a_3, a_4, a_5, a_6, a_7) \cap (a_7 \rightarrow a_1)) \leq (a_1 \rightarrow a_7) \end{aligned}$$

Sets of values V_2, \dots, V_6 are computed as follows:

$$\begin{aligned} V_2(a_1, a_3) &= \{E_2(a_1, a_2, a_3) | a_2\} \\ V_3(a_1, a_4) &= \{E_3(a_1, a_2, a_3, a_4) | a_2, a_3\} \\ V_4(a_1, a_5) &= \{E_4(a_1, a_2, a_3, a_4, a_5) | a_2, a_3, a_4\} \\ V_5(a_1, a_6) &= \{E_5(a_1, a_2, a_3, a_4, a_5, a_6) | a_2, a_3, a_4, a_5\} \\ V_6(a_1, a_7) &= \{E_6(a_1, a_2, a_3, a_4, a_5, a_6, a_7) | a_2, a_3, a_4, a_5, a_6\} \\ \text{For all } a_1, a_7 : \text{answer}(a_1, a_7) &\text{ follows from } V_6(a_1, a_7), a_7 \rightarrow a_1, \\ &\text{and } a_1 \rightarrow a_7 \end{aligned}$$

For example, $V_4(a_1, a_5)$ is the set of values $E_4(a_1, a_2, a_3, a_4, a_5)$ can have when a_2, a_3, a_4 range over all possibilities. If $\text{answer}(a_1, a_7)$ is true for all possible a_1 and a_7 , then 7-Go holds in the lattice, otherwise it fails.

The computation time is estimated as follows, where n is the number of nodes in the test lattice:

$$\begin{aligned} \text{Each } V_2(a_1, a_3) &\text{ can be found in } O(n) \text{ time; } O(n^3) \text{ total.} \\ \text{Each } V_3(a_1, a_4) &\text{ can be found in } O(n^2) \text{ time from } V_2; O(n^4) \text{ total.} \\ \text{Each } V_4(a_1, a_5) &\text{ can be found in } O(n^2) \text{ time from } V_3; O(n^4) \text{ total.} \\ &\vdots \end{aligned}$$

So the total time is $O(n^4)$.

The program is written so that it only has to compute additional “inner terms” to process the next n -Go equation. Remarkably, when a lattice does not violate any n -Go, the addition of new terms tends to converge to a fixed value rather quickly, meaning that V_n for $(n+1)$ -Go remains the same as V_{n-1} for n -Go. This almost always happens for $n < 10$, and when it does, we can terminate the algorithm and say with certainty that no further increase in n will cause an n -Go equation to fail in the lattice. (If it doesn’t happen, the program will tell us that, but such a case has so far not been observed. The program has an arbitrary cut-off point of $n = 100$, after which the algorithm will terminate. Observed runs have always either converged or failed far below this point, and in any case the cut-off can be increased with a parameter setting.) Convergence provides a proof that the entire class of Godowski equations (for *all* $n < \infty$) will pass in the lattice. Such a feat is not possible with ordinary lattice-checking programs, since an infinite number of equations would have to be tested.

When a lattice does violate some n -Go, that result tends to be found even faster, and the algorithm terminates, and the program tells us the first n at which an n -Go equation fails in the lattice. Since n -Go can be derived from $n+1$ -Go, failure is also implied for all greater n .

The success of `latticego.c` depends crucially on the structure of a particular representation of the n -Go equations, where variables appear only on one side of the equation and are localized to an adjacent pair of conjuncts in a chain of conjuncts. So far, efforts to adapt the approach to other equational families, such as the n OA (generalized orthoarguesian) laws, [7] haven’t been successful but are still being explored.

V. FINDING STATES ON FINITE LATTICES

It is possible to express the set of constraints imposed by states as a linear programming (LP) problem. Linear programming is used by industry to minimize cost, labor, etc., and many efficient programs have been developed to solve these problems, most of them based on the simplex algorithm.

We will examine a particular example in detail to illustrate how the problem is expressed. For this example we will consider a Greechie diagram with 3-atom blocks, although the principle is easily extended to any number of blocks.

If m is a state, then each 3-atom block with atoms a, b, c and complements a', b', c' imposes the following constraints:

$$\begin{aligned} m(a) + m(b) + m(c) &= 1 \\ m(a') + m(a) &= 1 \\ m(b') + m(b) &= 1 \\ m(c') + m(c) &= 1 \\ m(x) &\geq 0, \quad x = a, b, c, a', b', c' \end{aligned} \tag{24}$$

To obtain Eq. (24), note that in any Boolean block, $a \perp b \perp c \perp a$, so $m(a) = 1 - m(a') = 1 - m(b \cup c) = 1 - m(b) - m(c)$.

Let us take the specific example of the Peterson lattice, which we know does not admit a set of strong states. The Greechie diagram for this lattice, shown in Fig. 1, can be expressed with the textual notation

123,345,567,789,9AB,BC1,2E8,4FA,6DC,DEF.

(see Ref. [2]), where each digit or letter represents an atom, and groups of them represent blocks (edges of the Greechie diagram).

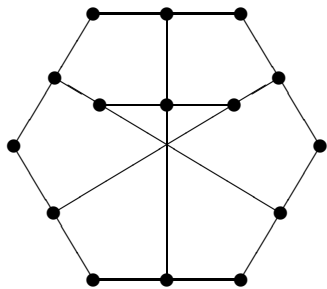


FIG. 1: Greechie diagram for the Peterson lattice.

Referring to the textual notation, we designate the atoms by $1, 2, \dots, F$ and their orthocomplements by $1', 2', \dots, F'$. We will represent the values of state m on the atoms by $m(1), m(2), \dots, m(F)$. This gives us the following constraints

for the 10 blocks:

$$\begin{aligned}
m(1) + m(2) + m(3) &= 1 \\
m(3) + m(4) + m(5) &= 1 \\
m(5) + m(6) + m(7) &= 1 \\
m(7) + m(8) + m(9) &= 1 \\
m(9) + m(A) + m(B) &= 1 \\
m(B) + m(C) + m(1) &= 1 \\
m(2) + m(E) + m(8) &= 1 \\
m(4) + m(F) + m(A) &= 1 \\
m(6) + m(D) + m(C) &= 1 \\
m(D) + m(E) + m(F) &= 1
\end{aligned}$$

In addition, we have $m(a') + m(a) = 1$, $m(a) \geq 0$, and $m(a') \geq 0$ for each atom a , adding potentially an additional $15 \times 3 = 45$ constraints. However, we can omit all but one of these since most orthocomplemented atoms are not involved this problem, the given constraints are sufficient to ensure that the state values for atoms are less than 1, and the particular linear programming algorithm we used assumes all variables are nonnegative. This speeds up the computation considerably. The only one we will need is $m(7) + m(7') = 1$ because, as we will see, the orthocomplemented atom $7'$ will be part of the full problem statement.

We pick two incomparable nodes, 1 and $7'$, which are on opposite sides of the Peterson lattice. (The program will try all possible pairs of incomparable nodes, but for this example we have selected a priori a pair that will provide us with the answer.) Therefore it is the case that $\sim 1 \leq 7'$. If the Peterson lattice admitted a strong set of states, for any state m we would have:

$$(m(1) = 1 \Rightarrow m(7') = 1) \Rightarrow 1 \leq 7'.$$

Since the conclusion is false, for some m we must have

$$\begin{aligned}
&\sim (m(1) = 1 \Rightarrow m(7') = 1) \\
\text{i.e. } &\sim (\sim m(1) = 1 \vee m(7') = 1) \\
\text{i.e. } &m(1) = 1 \ \& \ \sim m(7') = 1
\end{aligned}$$

So this gives us another constraint:

$$m(1) = 1;$$

and for a set of strong states to exist there must be some m such that

$$m(7) < 1.$$

So, our final linear programming problem becomes (expressed in the notation of the publicly available program `lp_solve[23]`):

```

min: m7';
m1 = 1;
m7 + m7' = 1;
m1 + m2 + m3 = 1;
m3 + m4 + m5 = 1;
m5 + m6 + m7 = 1;
m7 + m8 + m9 = 1;
m9 + mA + mB = 1;
mB + mC + m1 = 1;
m2 + mE + m8 = 1;
m4 + mF + mA = 1;
m6 + mD + mC = 1;
mD + mE + mF = 1;

```

which means “minimize $m(7')$, subject to constraints $m(1) = 1, m(7) + m(7') = 1, \dots$ ” The variable to be minimize, $m(7)$, is called the *objective function* (or “cost function”). When this problem is given to `lp_solve`, it returns an objective function value of 1. This means that regardless of m , the other constraints impose a minimum value of 1 on $m(7')$, contradicting the requirement that $m(7') < 1$. Therefore, we have a proof that the Peterson lattice does not admit a set of strong states.

The program `states.c` that we use reads a list of Greechie diagrams and, for each one, indicates whether or not it admits a strong set of states. The program embeds the `lp_solve` algorithm, wrapping around it an interface that translates, internally, each Greechie diagram into the corresponding linear programming problem.

VI. GENERATION OF MGES FROM FINITE LATTICES

When the linear programming problem in the previous section finds a pair of incomparable nodes that prove that the lattice admits no strong set of states, the information in the problem can be used to find an equation that holds in any OML admitting a strong set of states, and in particular in HL, but fails in the OML under test. Typically, an OML to be tested was chosen because it does not violate any other known HL equation. Thus, by showing an HL equation that fails in the OML under test, we will have found a new equation that holds in HL and is independent from other known equations.

The set of constraints that lead to the objective function value of 1 in our linear programming problem turns out to be redundant. Our algorithm will try to find a minimal set of hypotheses (constraints) that are needed. The equation-finding mode of `states.c` program incorporates this algorithm, which will try to weaken the constraints of the linear programming problem one at a time, as long as the objective function value remains 1 (as in the problem in the previous section). The equation will be constructed based on a minimal set of unweakened constraints that results.

The theoretical basis for the construction is described in the proof of Theorem 30 of Ref. [2]. Here, we will describe the algorithm by working through a detailed example.

Continuing from the final linear programming problem of the previous section, the program will test each constraint corresponding to a Greechie diagram block, i.e. each equation with 3 terms, as follows. It will change the right-hand side of the constraint equation from $= 1$ to ≤ 1 , thus weakening it, then it will run the linear programming algorithm again. If the weakened constraint results in an objective function value $m(7') < 1$, it means that the constraint is needed to prove that the lattice doesn't admit a strong set of states, so we restore the r.h.s. of that constraint equation back to 1. On the other hand, if the objective function value remains $m(7') = 1$ (as in the original problem), a tight constraint on that block is not needed for the proof that the lattice doesn't admit a strong set of states, so we leave the r.h.s. of that constraint equation at ≤ 1 .

After the program completes this process, the linear programming problem for this example will look like this:

```
min: m7';
m1 = 1;
m7 + m7' = 1;
m1 + m2 + m3 <= 1;
m3 + m4 + m5 = 1;
m5 + m6 + m7 <= 1;
m7 + m8 + m9 <= 1;
m9 + mA + mB = 1;
mB + mC + m1 <= 1;
m2 + mE + m8 = 1;
m4 + mF + mA <= 1;
m6 + mD + mC = 1;
mD + mE + mF <= 1;
```

Six out of the 10 blocks have been made weaker, and the linear programming algorithm will show that the objective function has remained at 1. We now have enough information to construct the MGE, which we will work with in the abbreviated form of a condensed state equation (Definition III.7).

1. Since $m(1) = 1$, the other atoms in the two blocks (3-term equations) using it will be 0. Thus $m(2) = m(3) = m(B) = m(C) = 0$.
2. For each of the four blocks that have $= 1$ on the r.h.s., we suppress the atoms that are 0 and juxtapose the remaining 2 atoms in each block. For example, in $m(3) + m(4) + m(5) = 1$, we ignore $m(3) = 0$, and collect the

atoms from the remaining two terms to result in 45 (4 juxtaposed with 5). Then we join all four pairs with + to build the l.h.s. form for the condensed state equation:

$$45 + 9A + E8 + 6D \quad (25)$$

3. For the r.h.s. of the equation, we scan the blocks with weakened constraints. From each block, we pick out and juxtapose those atoms that also appear on the l.h.s. and discard the others. For example, in $m(5)+m(6)+m(7) \leq 1, 5$ and 6 appear in Eq. (25) but 7 doesn't. Joining the juxtaposed groups with +, we build the r.h.s.:

$$56 + 89 + 4A + DE$$

Note that out of the 6 weakened constraints, 2 of them have no atoms at all in common with Eq. (25) and are therefore ignored.

4. Equating the two sides, we obtain the form of the condensed state equation:

$$45 + 9A + E8 + 6D = 56 + 89 + 4A + DE$$

5. Replacing the atoms with variables, the final condensed state equation becomes:

$$ab + cd + ef + gh = bg + fc + ad + he \quad (26)$$

6. Finally, the number of occurrences of each variable on must match on each side of the condensed state equation. In this example, that is already the case. But in general, there may be terms that will have to be repeated in order to make the numbers balance. An example with such "degenerate" terms is shown as Eq. (47) of Ref. [2].

Eq. (26) will be recognized, after converting it to an MGE, as the 4-Go equation, which as is well-known holds in all OMLs that admit a strong set of states but fails in the Peterson lattice Fig. 1. [7]

VII. SOLUTION TO AN OPEN PROBLEM

In Ref. [5], Mayet shows the following consequence of one of his equations (E_2^*) derived from Hilbert-space-valued states, and asks whether an OML exists in which it fails. The answer is negative.

Theorem VII.1 *The condition*

$$\begin{aligned} a_1 \perp b_1 \ \& \ a_2 \perp b_2 \ \& \ a_1 \perp a_2 \\ \Rightarrow \quad (a_1 \cup b_1) \cap (a_2 \cup b_2) &\leq b_1 \cup b_2 \cup (a_1 \cup a_2)' \end{aligned} \quad (27)$$

holds in all OMLs.

Proof. The following lemma follows using DeMorgan's law, the Foulis-Holland theorem (F-H; see e.g. Ref. [14, p. 25]), and the orthogonality hypothesis $a_2 \perp b_2$, respectively, for its three steps. The orthogonality hypotheses provide the commute relations needed for F-H.

$$\begin{aligned} b_2 \cup (a_1 \cup a_2)' &= b_2 \cup (a_1' \cap a_2') \\ &= (b_2 \cup a_1') \cap (b_2 \cup a_2') \\ &= (b_2 \cup a_1') \cap a_2'. \end{aligned} \quad (28)$$

From $(b_1 \cap a_1') \cup (b_2 \cup a_1') = a_1' \cup b_2$ (in any OL) and $(b_1 \cap a_1') \cup a_2' = a_2' \cup b_1$ (using $a_1 \perp b_1$), we have

$$(a_1' \cup b_2) \cap (a_2' \cup b_1) = ((b_1 \cap a_1') \cup (b_2 \cup a_1')) \cap ((b_1 \cap a_1') \cup a_2'). \quad (29)$$

The result then follows by applying the hypotheses, OL, Eq. 29, F-H, the hypothesis $a_1 \perp b_1$, and Eq. 28 to obtain the following steps, respectively:

$$\begin{aligned} (a_1 \cup b_1) \cap (a_2 \cup b_2) &\leq (a_2' \cup b_1) \cap (a_1' \cup b_2) \\ &= (a_1' \cup b_2) \cap (a_2' \cup b_1) \\ &= ((b_1 \cap a_1') \cup (b_2 \cup a_1')) \cap ((b_1 \cap a_1') \cup a_2') \\ &= (b_1 \cap a_1') \cup ((b_2 \cup a_1') \cap a_2') \\ &= b_1 \cup ((b_2 \cup a_1') \cap a_2') \\ &= b_1 \cup b_2 \cup (a_1 \cup a_2)'. \end{aligned}$$

□

VIII. CONCLUSION

In the previous sections we presented several results obtained in the field of Hilbert space equations based on the states defined on the space. The idea is to use classes of Hilbert lattice equations for an alternative representation of Hilbert lattices and Hilbert spaces of arbitrary quantum systems that might eventually enable a direct introduction of the states of the systems into quantum computers. In applications, infinite classes could then be “truncated” to provide us with finite classes of required length. The obtained classes would in turn contribute to the theory of Hilbert space subspaces, which so far is poorly developed.

In Sections III–VII we have considered three classes of Hilbert lattice equations based on the states defined on the lattice by means of Definition II.7 and specified in Section VII.

The algorithms and associated computer programs that were developed for this project were essential to its success. McKay’s dynamic programming algorithm for n -Go equations (Section IV), together with its quickly convergent behavior for large n , was particularly fortuitous. Without it, the authors see no apparent way that the independence of the MGE equations from all n -Go equations could be answered. At best, only empirical evidence pointing towards that answer could be accumulated, but that of course would not constitute a proof. Indeed, this problem had remained open for nearly 20 years since Mayet’s first publication [3] of these equations.

Thus, there is a strong motivation to find variants of McKay’s n -Go dynamic programming algorithm that could be more generally applied to other infinite families, in particular the n OA (generalized orthoarguesian) laws. [7] Assuming similar run-time behavior could be achieved, this would provide us with an extremely powerful tool that would let us test finite lattices against the family very quickly (instead of months or years of CPU time) as well as prove independence results for the entire infinite family of equations at once (if the valuation set rapidly converges to a final, fixed value with increasing n , as it does for n -Go). On the surface this appears to be quite a difficult problem, because unlike the special form in which the n -Go equations can be rewritten - so that most of the variables are localized to adjacent conjuncts - the known forms of the n OA laws have their variables distributed throughout their (very long) equations. So another approach would be to discover a new form of the n OA laws that better separates their variable occurrences in such a way that a variant of the n -Go dynamic programming algorithm might be applicable. Both of these approaches are being investigated by the authors.

The authors are unaware of any previous use of linear programming methods for finding states on a finite lattice and in particular (for the present study) determining whether the lattice admits a strong set of states. There appear to be relatively few programs that deal with states, and most of the finite lattice examples in the literature related to states were found by hand. A Pascal program written by Klaey [19] is able to find certain kinds of states on lattices, but for the strong set of states problem it is apparently able only to indicate “yes” (if a strong set of states was found) or “unknown” otherwise. The linear programming method provides a definite answer either way, in the predictable amount of time that the simplex algorithm takes to run. Finally, the linear programming problem itself (with redundant constraints weakened) provides us with the information we need to construct a new Hilbert lattice equation that fails in a given lattice not admitting a strong set of states.

Mayet’s recent and important E -equation results [5] provide us with powerful new method, the use of Hilbert-space-valued states, to find previously unknown families of equations that hold in Hilbert lattices. For further investigation of these equations, it will be highly desirable to have a program analogous to our *states.c* (which works with only real-valued states) that will tell us whether or not a finite lattice admits a strong set of Hilbert-space-valued states. This problem seems significantly harder than that of finding real-valued states, and possible algorithms for doing this are being explored by the authors.

The programs *latticego.c* and *states.c* described above can be downloaded from <http://us.metamath.org/#downloads>.

Acknowledgments

Supported by the Ministry of Science, Education, and Sport of Croatia.

-
- [1] M. Pavičić, *Quantum Computation and Quantum Communication: Theory and Experiments*, Springer, New York, 2005.
 - [2] M. Pavičić and N. D. Megill, *Quantum Logic and Quantum Computation*, in *Handbook of Quantum Logic*, edited by K. Engesser, D. Gabbay, and D. Lehmann, volume 2, pages 751–787, Elsevier, Amsterdam, 2006.
 - [3] R. Mayet, Varieties of Orthomodular Lattices Related to States, *Algebra Universalis* **20**, 368–396 (1985).

- [4] R. Godowski, Varieties of Orthomodular Lattices with a Strongly Full Set of States, *Demonstratio Math.* **14**, 725–733 (1981).
- [5] R. Mayet, Equations Holding in Hilbert Lattices, *Int. J. Theor. Phys.* **45**, 1216–1246 (2006).
- [6] L. Beran, *Orthomodular Lattices; Algebraic Approach*, D. Reidel, Dordrecht, 1985.
- [7] N. D. Megill and M. Pavičić, Equations, States, and Lattices of Infinite-Dimensional Hilbert Space, *Int. J. Theor. Phys.* **39**, 2337–2379 (2000).
- [8] M. Pavičić and N. D. Megill, *Is Quantum Logic a Logic?*, in *Handbook of Quantum Logic*, edited by K. Engesser, D. Gabbay, and D. Lehmann, volume 1, Elsevier, Amsterdam, 2006.
- [9] G. Birkhoff, *Lattice Theory*, volume XXV of *American Mathematical Society Colloquium Publications*, American Mathematical Society, New York, 2nd (revised) edition, 1948.
- [10] G. Birkhoff, *Lattice Theory*, volume XXV of *American Mathematical Society Colloquium Publications*, American Mathematical Society, Providence, Rhode Island, 3rd (new) edition, 1967.
- [11] M. Pavičić, Nonordered Quantum Logic and Its YES–NO Representation, *Int. J. Theor. Phys.* **32**, 1481–1505 (1993).
- [12] M. Pavičić, Identity Rule for Classical and Quantum Theories, *Int. J. Theor. Phys.* **37**, 2099–2103 (1998).
- [13] J. J. Zeman, Quantum Logic with Implications, *Notre Dame J. Formal Logic* **20**, 723–728 (1979).
- [14] G. Kalmbach, *Orthomodular Lattices*, Academic Press, London, 1983.
- [15] G. Kalmbach, *Measures and Hilbert Lattices*, World Scientific, Singapore, 1986.
- [16] G. Kalmbach, *Quantum Measures and Spaces*, Kluwer, Dordrecht, 1998.
- [17] M. J. Mączyński, Hilbert Space Formalism of Quantum Mechanics without the Hilbert Space Axiom, *Rep. Math. Phys.* **3**, 209–219 (1972).
- [18] B. D. McKay, N. D. Megill, and M. Pavičić, Algorithms for Greechie Diagrams, *Int. J. Theor. Phys.* **39**, 2381–2406 (2000).
- [19] M. Kläy, *Stochastic Models on Empirical Systems, Empirical Logic and Quantum Logics, and States on Hypergraphs*, PhD thesis, University of Bern, Faculty of Natural Science, Fischer Druck, Münsingen, 1985.
- [20] E. G. Beltrametti and G. Cassinelli, *The Logic of Quantum Mechanics*, Addison-Wesley, 1981.
- [21] S. S. Holland, JR., Orthomodularity in Infinite Dimensions; a Theorem of M. Solèr, *Bull. Am. Math. Soc.* **32**, 205–234 (1995).
- [22] For additional definitions of the terms used in this section see Refs. [7, 15, 20, 21].
- [23] Available at <http://groups.yahoo.com/group/lp.solve/files/> (September 2006).